On Relational Language Translation

Adam Dudáš
Department of Computer Science
Matej Bel University
Banská Bystrica, Slovakia
adam.dudas@umb.sk

Daniel Demian
Department of Computer Science
Matej Bel University
Banská Bystrica, Slovakia
daniel.demian@student.umb.sk

Jarmila Škrinárová
Department of Computer Science
Matej Bel University
Banská Bystrica, Slovakia
jarmila.skrinarova@umb.sk

***Abstract*—Relational database model, as one of the most commonly used database models, is defined by the structure used for the storing of data, methods of data manipulation and so called data integrity. In this paper, we are focusing on the methods and processes of manipulating data stored in the database with the use of relational algebra, relational calculus and Structured Query Language. The paper presents the design, implementation and experimental evaluation of a software tool that can be used for automatic translation between the mentioned relational languages themselves and restricted natural language with the integrated automatic optimization of queries in relational algebra.**

***Index Terms*—relational databases, relational algebra, relational calculus, SQL, translating**

I. Introduction

Relational database model has been the cornerstone of effective data storing for more than fifty years. Any data model is defined by number of basic properties [1]:

- data structure in which the data are stored,
- data integrity used for assurance of data accuracy and consistency,
- methods and processes of manipulating data stored in the database.

Presented work focuses on the latter – manipulation of data in the database – in the context of relational databases. To work with such data, it is necessary to have a defined set of operations suitable for various tasks within a given database. In our work we will consider the following languages for data manipulation based on query processing [2], [3], [4]:

- formal relational languages – relational algebra and relational calculus,
- Structured Query Language (SQL).

The paper presents the design, implementation and experimental evaluation of a software tool that can be used for automatic translation between the mentioned formal relational languages, SQL and restricted natural language. This tool also contains automatic optimization of queries in relational algebra. As far as natural language is concerned, we are working with Slovak language, which was preferred over English mainly due to the potential use of the proposed software tool in the educational process at universities in Slovakia, which teach the subjects related to the relational databases in Slovak.

The work presented in this paper consists of the following sections:

- Section II contains analysis of works related to the relational database model, formal relational languages and applications with the similar objective to the presented software tool.
- In the section III, we briefly present relational languages used in our work, specifically relational algebra, tuple relational calculus and Structured Query Language.
- Sections IV and V present design, implementation and experimenal evaluation of the proposed tool, its functions and functionality.

II. Related works

Even though various types of database systems are slowly suppressing traditional use of relational databases, the research in the area of relational databases and use of these databases is still relevant [5], [6]. Formal relational languages and Structured Query Language are parts of several concepts and subsystems of relational databases, e.g. optimization of query processing in the database system [7].

In [1] authors discuss the auto–indexing methods provided by relational database systems and highlight limitations of these approaches. Authors also propose original techniques for removing of the impact of peaks caused by adding new queries to the system, to which no suitable index is present. The work presented in [8] is focused on storage principles and proposal of novel methods for effective data block location and identification if no suitable index for the query is present in the system. This approach leads to optimization of the performance of the whole system and lowering of the processing time and costs. The paper [9] deals with the relational database system architecture and proposes novel techniques for optimizing data location and access to the tuples in order to retrieve relevant data effectively.

Authors of [4] implement a hybrid query optimizer that intelligently and transparently combines both binary and multi–way joins within the same query processing plan. Demonstration of this approach shows, that presented methods outperforms existing systems when worst–case optimal joins are beneficial while sacrificing no performance when they are not.

Since the objective of the presented paper is design and implementation of translator for relational languages with the potential use in the area of education, we also present recent research related to this area.

979-8-3503-1034-4/22/$31.00 ©2022 IEEE

In [10] the authors present a software module developed for the course Semantics of Programming Languages. This software was created as a part of the intended comprehensive software package to simplify and make the teaching of formal principles in theoretical computer science more attractive. Authors of [11] present an example of use of blended learning education for the university courses with the use of specifically designed tools for Fuzzy sets courses. In the [12], [13] authors present tools created for the needs of analysis of edge coloring of graphs which are based on visualization of edge coloring. These software tools are currently used in the context of courses and workshops focused on graph problems.

III. Relational languages

This work focuses on translation between various languages used to retrieve data from relational databases – the action widely known as querying [5]. We consider three relational languages – relational algebra, relational calculus and Structured Query Language.

A. Relational algebra

Relational algebra is a formal procedural query language designed to work with relational databases, which can be classied as prescriptive [2]. A query written using relational algebra can be generalized as follows:

$$operation_{predicate}(R)$$

where $predicate$ denotes set of attributes, conditions or functions and R is one or more relations of given database.

Basic operations of relational algebra are:

- projection (π) – with the use of projection, we are able to compute relation, which contains set of attributes (A) chosen from the input relation (R) specified in the predicate of the operation. The projection can also be used as a way of changing order in which the attributes are presented.

$$\pi_A(R) \tag{1}$$

- selection (σ) – can be used for generating relation, which consists of tuples of input relation(s) (R) meeting the conditions specified in the predicate of the operation (C). The conditions can be composed of one or more expressions interconnected with the use of conjunction or disjunction.

$$\sigma_C(R) \tag{2}$$

- join ($\bowtie$) – serves as a way of joining relations which contain common attribute – implemented as standard primary and foreign key relationship. In the formal notation (3) there are two input relations R_1, R_2 joined with the use of primary key of the relation R_1 ($R_1.PK$) and foreign key of relation R_2 ($R_2.FK$).

$$R_1 \bowtie_{R_1.PK=R_2.FK} R_2 \tag{3}$$

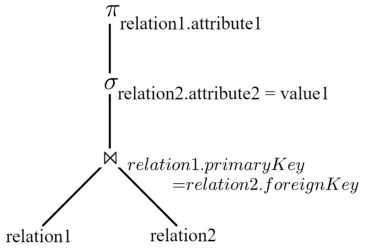

Fig. 1. Example of relational algebra query written in the form of expression tree

Other relational algebra operations are aggregation and grouping (α), renaming (ρ) and all set operations, e.g. union ($\cup$), intersection ($\cap$), cartesian product ($\times$) [3].

Since relational algebra is prescriptive language, the order of specified operations changes the data processing procedure in the query. In order to use query with (pseudo) optimal order of operations, relational algebra uses two simple optimization operations – push down projection and push down selection. Both operations are realted to the expression tree of the query – relational algebra query written in the structure of a graph (tree) that is branched by relation joins (see Figure 1). These optimization operations push their respective relational operations (projection, selection) to the lowest possible position in the expression tree (closest to the leaves of the tree). Therefore optimiziation operations ensure lowering of the amount of data in the join section of the expression in the following way:

- push down projection – lower number of attributes as an input for joining of relations,
- push down selection – lower number of tuples (entities) as an input for joining of relations.

For the examples of queries written in the form of relational algebra see Section V.

B. Tuple relational calculus

There are two types of relational calculus most commonly used – tuple relational calculus and domain relational calculus. We are focusing on the former. Similar to relational algebra, tuple relational calculus is a formal procedural query language designed to work with relational databases. In the case of relational calculus, its use is less prescriptive and more descriptive [2].

In general, tuple relational calculus can be formaly nottated as follows:

$$\{N|f(N)\}$$

where N is a tuple variable that describes the output of relational calculus operations and $f(N)$ is a function that describes the values that the variable N can acquire. The output of the operation consists of tuples of values that meet the conditions defined in $f(N)$.

Queries written in the tuple relational calculus are composed of combination of following constructs:

- $R \in REL$ – affiliation of the tuple denoted by the variable R to the relation called REL.
- $p \wedge q$, $p \vee q$, ... – operations of relational calculus connected by conjunction or disjunction. In this case, the operations are simple conditions for values that can be acquired by the output tuple variable.
- $\exists R(f(R))$ – existence of tuple(s) denoted by R which satisfies $f(R)$.

For the examples of queries written in the form of tuple relational calculus see Section V.

C. Structured Query Language

SQL was designed as a database language for relational databases in mid–1970s. This language was standardized in 1986 and has undergone a number of modifications and updates since. From the point of view of structure of SQL itself, we can identify number of sub–languages:

- Data Definition Lnaguage,
- Data Manipulation Language,
- Data Query Language,
- Data Control Language,
- Transaction Control Language.

In the context of the presented research, the Data Query Language is of interest – the operations of this language are used to perform queries on data stored in the relational database.

IV. Translator for relational languages

As described in the introduction of the paper, our main objective is to present translator for relational languages based on the following criteria:

- Designed tool needs to be able to translate between relational algebra, tuple relational calculus and Structured Query Language in all directions.
- The application needs to be able to translate from restricted Slovak language into abovementioned relational languages.
- Other than translating from and to relational algebra, we require possibility of query optimization in this language.

These criteria yield a few remarks concerning the translation itself.

As described in the Section III.B, tuple relational calculus uses tuple variables as a way to mark attributes affiliation to its respective relation. For the sake of better readability of the tuple relational calculus expression these tuple variables should be named after the first letter of the title of its relation. In the case that more than one relations in the expression have the title starting with the same letter, translator should use shortest possible substring of the original relation title to name the tuple variables.

When working with more than one relation, we need to use full name of an attribute. In the relational algebra and SQL the name bears the shape of *relation_name.attribute_name*. In the relational calculus the full name of the attribute is formed with the use of tuple variable – *tuple_variable_name.attribute_name*.

$$
\begin{array}{l}
T \rightarrow StabuľkySV|StabulkySV|StabuľkaSV|StabulkaSV|\epsilon \\
S \rightarrow aS|bS|...|zS|AS|BS|...|ZS|0S|1S|...|9S|\text{“”}|\_ \\
V \rightarrow vyberteN|vyberN|výberN \\
N \rightarrow S.P|S,N \\
P \rightarrow kdeSO|T \\
O \rightarrow sarovnaH|sarovnáH|jerovneH|jemenšiH2|jemensiH2|jemensieH2| \\
jemenšieH2|jemenejH2|jeväčšiH2|jevacsiH2|jeväčšieH2|jevacsieH2 \\
H \rightarrow SaSO|S.T \\
H2 \rightarrow akoH|H
\end{array}
$$

Fig. 2. CFG approximating restricted language used in the translation of queries

When translating from natural language the language used as an input for the translation to relational languages needs to be – at least partially – restricted [14]. We approximate the context free gramar (CFG) for the proposed language in the Figure 2.

With the use of these criteria, we built desktop application in the Python language using the $tkinter$ library. We divided the program into classes according to the language from which we translate. First, a text string from the input is loaded, which is then split into an array of words using the $split()$ method - space is used as a separator. Subsequently, we send this field to helper methods that return translated strings. In the $translateRa$ method, these strings are concatenated into one and used as an output within the GUI.

V. Experimental evaluation of the translator for relational languages

Based on the principles of relational language translation described in the Sections III and IV, we are presenting the queries translated with the use of implemented translation tool. Basic translating of relational languages is presented in the following way:

- The Table I presents translating queries of various complexity from Structured Query Language to relational algebra and relational calculus.
- In the Table II, we present translating queries of various complexity from relational algebra to relational calculus and SQL.
- The Table III presents translating queries of various complexity from relational calculus to SQL and relational algebra.

Other than basic translating from and to various relational languages presented application offers two other functionalities:

- The Table IV describes translation of queries of various complexity from natural language to relational languages. Natural language–wise we are working with Slovak language, which was preferred over English mainly due to the potential use of the proposed software tool in the local educational process. We are also presenting English version of the input queries:
  - Choose *name* and *age* from the table *person*.

  - Select *name* and *age* from the table *person* where *name* equals to *Daniel* and the *age* is higher than *20* and lower than *30*.
  - Select *name* from the table *person*. From the table *programme* select *abbreviation* where *abbreviation* equals *AIN*.
  - We need to select *name*, *surname* from the table *person* where *surname* equals to *Smith*, from the table *programme*, we need to select *abbreviation* which equals *AIN*. Finally, from the table *school* select *title*.
- In the Table V, we present optimization of relational algebra queries which vary in the complexity. This optimization is based on the operations described in the Section III.A of the paper.

The first run of experimental testing of the presented application revealed minor problems in translating. Some of these problems were only superficial, e.g. redundant use of spaces, and some of the problems were functional, e.g. placement of commas in the relationala algebra and SQL queries. These problems have been fixed after modifying the translation tool.

VI. Conclusion

This paper focuses on the manipulation of data in the context of relational databases. To work with such data, it is necessary to have a defined set of operations suitable for various tasks within a given database. In our case we used query processing in the languages of relational algebra, tuple relational calculus and Structred Query Language.

We present software tool that can be used for automatic translation between the mentioned formal relational languages – relational algebra and tuple relational calculus –, SQL and restricted natural language. Since the software tool has the potential to be used in the educational process at universities in Slovakia, which teach the subjects related to the relational databases in Slovak, we focused on the translation from restricted Slovak language. The translation from and to all of the mentioned languages was experimentally tested with the use of variously compley queries. These translations are presented in the Tables I – IV.

In the Table V, we present other functionality of implemented tool – automatic optimization of queries in relational algebra.

Future work related to the created translator for relational languages contains adding interactive canvas for visualization of relational algebra queries in the form of expression tree in order to manually optimize given query with the use of push down selection and push down projection. Other objective in the future work is to expand the application with translation from English language, translation from spoken language in the direction towards low–code and no–code programming in the context of databases.

Acknowledgment

Computing was performed in the High Performance Computing Center of the Matej Bel University in Banska Bystrica using the HPC infrastructure acquired in project ITMS 26230120002 and 26210120002 (Slovak infrastructure for high-performance computing) supported by the Research & Development Operational Programme funded by the ERDF.

The research was partially supported by the grant of The Ministry of Education, Science, Research and Sport of Slovak Republic - Implementation of new trends in computer science to teaching of algorithmic thinking and programming in Informatics for secondary education, project number KEGA 018UMB-4/2020.

The support of Advtech_AirPollution project (Applying some advanced technologies in teaching and research, in relation to air pollution, 2021-1-RO01-KA220-HED-000030286) funded by European Union within the framework of Erasmus+ Program is gratefully acknowledged.

References

[1] Kvet, M., Matiasko, K. Analysis of current trends in relational database indexing. (2020) Proceedings of 2020 International Conference on Smart Systems and Technologies, SST 2020, art. no. 9264034, pp. 109-114. DOI: 10.1109/SST49455.2020.9264034

[2] E. F. Codd. A relational model of data for large shared data banks. Commun. ACM 13, 6, 1970, 377–387. DOI:10.1145/362384.362685

[3] A. Klug. Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions. J. ACM 29, 3, 1982, pp. 699–717. DOI: 10.1145/322326.322332

[4] Michael Freitag, Maximilian Bandle, Tobias Schmidt, Alfons Kemper, and Thomas Neumann. Adopting worst-case optimal joins in relational database systems. Proc. VLDB Endow. 13, 12, 2020, pp. 1891–1904. DOI: 10.14778/3407790.3407797

[5] Kvet, M., Papan, J. The Complexity of the Data Retrieval Process Using the Proposed Index Extension. (2022) IEEE Access, Vol.10, pp. 46187-46213. DOI: 10.1109/$ACCESS$.2022.3170711

[6] Tareq, M., Sundararajan, E.A., Harwood, A., Bakar, A.A. A Systematic Review of Density Grid-Based Clustering for Data Streams. (2022) IEEE Access, 10, pp. 579-596. DOI: 10.1109/$ACCESS$.2021.3134704

[7] Elena Botoeva, Diego Calvanese, Benjamin Cogrel, and Guohui Xiao. Formalizing MongoDB queries. Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017.

[8] Kvet, M., Matiasko, K. Data block and tuple identification using master index. (2020) Sensors (Switzerland), 20 (7), art. no. 1848. DOI: 10.3390/s20071848

[9] Kvet, M., Matiasko, K. Efficiency of the relational database tuple access. (2019) INFORMATICS 2019 - IEEE 15th International Scientific Conference on Informatics, Proceedings, art. no. 9119325, pp. 231-236. DOI: 10.1109/$Informatics$47936.2019.9119325

[10] Steingartner, W. Compiler Module of Abstract Machine Code for Formal Semantics Course. (2021) SAMI 2021 - IEEE 19th World Symposium on Applied Machine Intelligence and Informatics, Proceedings, art. no. 9378696, pp. 193-199. DOI: 10.1109/$SAMI$50585.2021.9378696

[11] Michalikova, A., Povinsky, M. Blended Learning as a Way of Teaching in the Pandemic Period. (2020) ICETA 2020 - 18th IEEE International Conference on Emerging eLearning Technologies and Applications, Proceedings, art. no. 9379249, pp. 464-469. DOI : 10.1109/$ICETA$51985.2020.9379249

[12] Dudas, A., Skrinarova, J., Kiss, A. On Graph Coloring Analysis through Visualization. (2021) International Conference on Information and Digital Technologies 2021, IDT 2021, art. no. 9497524, pp. 65-72. DOI: 10.1109/IDT52577.2021.9497524

[13] Dudas, A., Janky, J., Skrinarova, J. Web Application for Graph Visualization Purposes. (2020) ICETA 2020 - 18th IEEE International Conference on Emerging eLearning Technologies and Applications, Proceedings, art. no. 9379200, pp. 90-95. DOI: 10.1109/$ICETA$51985.2020.9379200

[14] Hudec, M., Smutny, Z. Ambient Intelligence System Enabling People With Blindness to Develop Electrotechnical Components and Their Drivers. (2022) IEEE Access, Vol.10, pp. 8539-8565. DOI: 10.1109/$ACCESS$.2022.3144109

TABLE I
TRANSLATING FROM SQL TO RELATIONAL ALGEBRA AND RELATIONAL CALCULUS

SQL	**Relational algebra**	**Tuple relational calculus**
SELECT name, surname FROM personalInfo WHERE city = “New York“;	π name, surname (σ city = ”New York” (personalInfo))	{N | $\exists$ P $\in$ personalInfo (P.city = ”New York” $\wedge$ N.name = P.name $\wedge$ N.surname = P.surname)}
SELECT name, surname FROM personalInfo WHERE birthyear $>$ 1990 AND birthyear $<$ 1996;	π name, surname (σ birthyear $>$ 1990 AND birthyear $<$ 1996 (personalInfo))	{N | $\exists$ P $\in$ personalInfo (P.birthyear $>$ 1990 $\wedge$ P.birthyear $<$ 1996 $\wedge$ N.name = P.name $\wedge$ N.surname = P.surname)}
SELECT personalInfo.name, personalInfo.surname, programme.title FROM personalInfo INNER JOIN programme ON personalInfo.programme_id = programme.id;	π personalInfo.name, personalInfo.surname, programme.title (personalInfo $\bowtie_{personalInfo.programme\_id=programme.id}$ programme)	{N | $\exists$ P $\in$ personalInfo $\wedge$ $\exists$ Pr $\in$ programme (P.programme_id = Pr.id $\wedge$ N.name = P.name $\wedge$ N.surname = S.surname $\wedge$ N.title = Pr.title)}
SELECT person.name, school.title, city.name FROM person INNER JOIN city ON person.city = city.id INNER JOIN school ON person.school = school.id WHERE person.name = ”Daniel” AND city.name = ”New York”	π person.name, school.title, city.name (σ person.name = ”Daniel” AND city.name = ”New York” (person $\bowtie_{person.city=city.id}$ city $\bowtie_{person.school=school.id}$school))	{N | $\exists$ P $\in$ person $\wedge$ $\exists$ C $\in$ city $\wedge$ $\exists$ S $\in$ school (P.city = C.id $\wedge$ P.school = S.id $\wedge$ P.name = ”Daniel” $\wedge$ C.name = ”New York” $\wedge$ N.name = P.name $\wedge$ N.title = S.title $\wedge$ N.name = C.name)}

TABLE II
TRANSLATING FROM RELATIONAL ALGEBRA TO RELATIONAL CALCULUS AND SQL

Relational algebra	**Tuple relational calculus**	**SQL**
π title (σ aid = 100 (book))	{N | $\exists$ B $\in$ book (B.aid = 100 $\wedge$ N.title = K.title)}	select title from book where aid = 100
π book.title (σ author.surname = ”McCarthy” (book $\bowtie_{book.aid=author.aid}$ author))	{N | $\exists$ B $\in$ book $\wedge$ $\exists$ A $\in$ author (B.aid = A.aid $\wedge$ A.surname = ”McCarthy” $\wedge$ N.title = B.title)}	select book.title from book inner join author on book.aid = author.aid where author.surname = ”McCarthy”
π book.title (σ author.name = “H.P.“ AND author.surname = “Lovecraft“ (book $\bowtie_{book.aid=author.aid}$ author))	{N | $\exists$ B $\in$ book $\wedge$ $\exists$ A $\in$ author (B.aid = A.aid $\wedge$ A.name = “H.P.“ $\wedge$ A.surname = “Lovecraft“ $\wedge$ N.title = K.title)}	select book.title from book inner join author on book.aid = author.aid where author.name = “H.P.“ AND author.surname = “Lovecraft“
π genre.title (σ author.nationality = “England“ AND book.year $>$ 2000 (genre $\bowtie_{genre.gid=book.gid}$ book $\bowtie_{book.aid=author.aid}$ author))	{N | $\exists$ G $\in$ genrer $\wedge$ $\exists$ B $\in$ book $\wedge$ $\exists$ A $\in$ author (G.gid = B.gid $\wedge$ B.aid = A.aid $\wedge$ A.nationality = “England“ $\wedge$ B.year $>$ 2000 $\wedge$ N.title = G.title)}	select genre.title from genre inner join book on genre.gid = book.gid inner join author on book.aid = author.aid where author.nationality = “England“ AND book.year $>$ 2000

TABLE III
TRANSLATING FROM RELATIONAL CALCULUS TO SQL AND RELATIONAL ALGEBRA

Tuple relational calculus	**SQL**	**Relational algebra**
{N | ∃ C ∈ car (N.model = C.model ∧ N.mileage = C.mileage)}	select model, mileage from car	π model, mileage (car)
{N | ∃ C ∈ car (C.dom= 25.11.2020 ∧ N.model = C.model)}	select model from car where dom = 25.11.2020	π model (σ dom = 25.11.2020 (car))
{N | ∃ R ∈ res ∧ ∃ O ∈ order (R.oid = O.oid ∧ O.name = "Novák" ∨ O.name = "Nováková" ∧ N.id = R.id)}	select res.id from res inner join order on res.oid = order.oid where order.name = "Novák" OR order.name = "Nováková"	π res.id (σ order.name = "Novák" OR order.name = "Nováková" (res ⋈$_{res.oid=order.oid}$ order))
{N | ∃ O ∈ order ∧ ∃ R ∈ res ∧ ∃ C ∈ car (O.id = R.oid ∧ R.cid = C.cid ∧ C.dom > 2000 ∧ N.name = O.name ∧ N.model = C.model)}	select order.name, car.model from order inner join res on order.id = res.oid inner join car on res.cid = car.cid where car.dom > 2000	π order.name, car.model (σ car.dom > 2000 (order ⋈$_{order.id=res.oid}$ rez ⋈$_{res.cid=car.cid}$ car))

TABLE IV
TRANSLATING FROM NATURAL LANGUAGE TO SQL, RELATIONAL ALGEBRA AND RELATIONAL CALCULUS

Natural language (Slovak)	**SQL**	**Relational algebra**	**Tuple relational calculus**
Z tabulky osoba vyber meno, vek.	select meno, vek from osoba	π meno, vek (osoba)	{N | ∃ O ∈ osoba (N.meno = O.meno ∧ N.vek = O.vek)}
Z tabulky osoba vyberte meno vek kde meno sa rovná Daniel a vek je vačší 20 a vek je menej 30.	select meno, vek from osoba where meno = Daniel AND vek > 20 AND vek < 30	π meno vek (σ meno = Daniel AND vek > 20 AND vek < 30 (osoba))	{N | ∃ O ∈ osoba (O.meno = Daniel ∧ O.vek > 20 ∧ O.vek < 30 ∧ N.meno = O.meno ∧ N.vek = O.vek)}
Z tabulky osoba vyber meno. Z tabulky odbor vyber skratka kde skratka sa rovná AIN.	select osoba.meno, odbor.skratka from osoba inner join odbor on osoba.odbor = odbor.id where odbor.skratka = AIN	π osoba.meno, odbor.skratka (σ odbor.skratka = AIN (osoba ⋈$_{osoba.odbor=odbor.id}$ odbor))	{N | ∃ O ∈ osoba ∧ | ∃ Od ∈ odbor (O.odbor = Od.id ∧ Od.skratka = AIN ∧ N.meno = O.meno ∧ N.skratka = Od.skratka)}
Potrebujeme z tabulky osoba vybrať meno, priezvisko kde priezvisko sa rovná Šmith a vek je menší ako 25. Ďalej z tabulky odbor potrebujeme vybrať skratka kde skratka sa rovná AIN. Nakoniec z tabulky skola vyberte nazov.	select osoba.meno, osoba.priezvisko, odbor.skratka, skola.nazov from osoba inner join odbor on osoba.odbor = odbor.id inner join skola on odbor.skola = skola.id where osoba.priezvisko = Smith AND osoba.vek < 25 AND odbor.skratka = AIN	π osoba.meno, osoba.priezvisko, odbor.skratka, skola.nazov (σ osoba.priezvisko = Smith AND osoba.vek < 25 AND odbor.skratka = AIN (osoba ⋈$_{osoba.odbor=odbor.id}$ odbor ⋈$_{odbor.skola=skola.id}$ skola))	{N | ∃ O ∈ osoba ∧ ∃ Od ∈ odbor ∧ ∃ S ∈ skola (O.odbor = Od.id ∧ Od.skola = S.id ∧ O.priezvisko = Smith ∧ O.vek < 25 < Od.skratka = AIN ∧ N.meno = O.meno ∧ N.priezvisko = O.priezvisko ∧ N.skratka = Od.skratka ∧ N.nazov = S.nazov)}

TABLE V
OPTIMIZATION OF RELATIONAL ALGEBRA QUERIES

Relational algebra	**Optimized relational algebra query**
π publisher.title (σ author.name = "Bram" AND author.surname = "Stoker" (publisher ⋈$_{publisher.pid=book.pid}$ book ⋈$_{book.aid=author.aid}$ author))	(π publisher.title (publisher)) ⋈$_{publisher.pid=book.pid}$ (book)) ⋈$_{book.aid=author.aid}$ (σ author.name = "Bram" AND author.surname = "Stoker" (author))
π genre.title (σ author.nationality = "England" AND book.year > 2000 (genre ⋈$_{genre.gid=book.gid}$ book ⋈$_{book.aid=author.aid}$ author))	(π genre.title (genre)) ⋈$_{genre.gid=book.gid}$ (σ book.year > 2000 (book)) ⋈$_{book.aid=author.aid}$ (σ author.nationality = "England" (author))
π person.name, person.surname, programme.abbrv, school.title (σ person.surname = Smith AND person.age < 25 AND programme.abbrv = AIN (person ⋈$_{person.pid=programme.id}$ programme ⋈$_{programme.sid=school.id}$ school))	(π person.name, person.surname (σ person.surname = Smith AND person.age < 25 (person))) ⋈$_{person.pid=programme.id}$ (π programme.abbrv (σ programme.abbrv = AIN (programme))) ⋈$_{programme.sid=school.id}$ (π school.title (school))
π order.name, car.model (σ car.dom > 2000 (order ⋈$_{order.id=res.oid}$ res ⋈$_{res.cid=car.cid}$ car))	(π order.name (order)) ⋈$_{order.id=res.oid}$ (res) ⋈$_{res.cid=car.cid}$ (π car.model (σ car.dom > 2000 (car)))